Waveform Averaging

Local application WAVG

Robert Goodwin

Tue, Nov 25, 2008

To get a better value for a waveform than one that is sampled at one time, we can average the readings of a portion of the waveform. In the Linac, the beam pulse may be only 10–30 μs wide, and a waveform may be measured at 20 MHz, giving up to 600 points. To average over 5 μs, say, we would need to average 100 data points. This simple LA does this.

Parameters layout

Field		*Size*	*Meaning*
`ENABLE`	`B`	2	Usual local application enable Bit#
`EVENT`		2	Clock event#
`WAVE`	`C`	2	First waveform-related Chan#
`NWAVES`		2	#waveforms in consecutive channels
`TIME`	`C`	2	Initial time in average range, Chan#
`RANGE`	`C`	2	Average time range, Chan#
`AVG`	`C`	2	First average result Chan#

The `EVENT` parameter selects those cycles on which the waveform averaging is to take place. One may need to average over a different portion of the waveform depending on the type of beam cycle, as indicated by a clock event. Although the event number is represented as a raw parameter value here, it could be easily changed to be the channel`#` that holds the event`#`.

The `TIME` and `RANGE` parameters specify the portion of the waveform to be averaged. For the LA to interpret the values carried by these two channels, we should assume some units for that raw data. Assume the units are 0.1 μs. The LA can determine where to find the data points to be averaged by knowing the digitization rate, which can be obtained by examination of the Quickr digitizer control register, located at offset `0x26` from the base of the Quickr registers. This base address can be found from the `CINFO` table entry. Bits 10–8 of the digitizer control register specify a power of two by which 20 MHz should be reduced in order to obtain the rate. In this way, `000` means 20 MHz, `001` means 10 MHz, `010` means 5 MHz, etc.

For each channel in the range indicated by the `WAVE` and `NWAVES` parameters, use the `CINFOentry` function to find and collect a copy of the `CINFO` table entry for the Quickr digitizer. The entry includes the address of the register base and also the base address of the waveform itself. We can just do this once, because the `WAVE` and `NWAVES` parameters are not expected to be changed during operation; neither is the `AVG` parameter. But the *values* that may be changed during operation of `WAVG` are the `TIME` and `RANGE` values—not the channel# parameters that point to these values. Using this option, one can make a plot showing the dependence of the average results on the portion of the waveform being averaged, for example.

The calculation of average waveform data points is done simply on the raw data. This means that the scaling of the average results should be the same as a sampled waveform data point.

Post-implementation

Under test, the program acts as described above. One surprise was that the time to access the data points in the Quickr waveform memory is about 2 μs/point. Calculating averages over 100 points for 8 waveforms therefore takes about 1.6 ms. (The time to execute the rest of the logic in the LA is trivial, given the fast PowerPC processor.)

To investigate whether another method of access can be found, two diagnostic choices were included in the code, each selected by a bit in the `ERRCODE` parameter, which is a diagnostic found in that last (10th) parameter word (not shown above). Setting bit #0 in this parameter word opts for accessing the data words using non-aligned 32-bit accesses. (Each such access is actually broken down into two 16-bit accesses by the hardware.) Setting bit #1 in the parameter opts for stopping further digitizations before accessing the data words, in the hope that this would reduce access time as a result of no contention required with the hardware accesses. Neither option helped.

Another issue that arose during testing is that the time needed by the hardware to record the digitized values into its waveform memory is about 100 ms for the maximum 16384 points (for each of the 8 channels). If we only wait for 10 ms before sampling the data points, about 1600 points will have been stored. At the 20 MHz rate, this covers only 80 μs of waveform. Perhaps this is enough. In the front ends using this LA, there is already a need to wait for replies to arrive from several SRMs that are connected via arcnet. The time needed for that is about 10 ms. The local applications are not invoked until after the data pool has been filled, which certainly includes data from the SRMs.